Final Team Reflection

Team 22

Customer Value and Scope

the chosen scope of the application under development including the priority of features and for whom you are creating value

- **A:** In the beginning of the course, we focused a lot on what features could be implemented in parallel. After meetings with our supervisor we realized that our main focus should be to implement what gives the most value to our stakeholders. Our initial thought was to include a menu where we easily could work on different parts of the code simultaneously. This led to the app having unfinished segments and made it look overall less clean.

  The first thing we implemented was an interactive map using the Google Maps API, since this is the most fundamental part of the application. Later on we created the menu with different buttons where the user would be able to navigate to the different segments. Initially we also thought that a QR code scanner would be used to unlock an umbrella, which made us design the app with this in mind. However, later on in the development we realized that this may not be required for the hardware and therefore it went from a highly prioritized user story to one of the less prioritized.

  Once we had these most foundational building blocks in place, we began developing features such as being able to rent an umbrella, cancel that rental and see information about it. Our stakeholders also thought it was of great value to be able to see your current location as well as finding directions to a specific stand.

  After this, our *minimal viable product* was almost done, but before the demonstration of the app we focused on creating maximal value for our stakeholders as well as the end user. This includes that a user would be able to complete a rental without any failures. Our stakeholders also requested a user story to store all rentals in the database with the argument that they found it important to be able to get how much earnings they get and know how many people use the application.

- **B:** In future projects, we are going to immediately get our user stories from what gives the stakeholders the most value, rather than focusing on implementing as many features as possible first.

  Our workload most weeks was definitely manageable and not too intense based on the numbers in our KPI evaluations. The evaluations also show that we almost always managed to reach our sprint goals. The workload during the first sprint was not too intense, which made us believe we could add more user stories for the next sprint.

This however led to not all user stories being finished, and perhaps a little rushed, which is something we want to avoid.

- **A → B:** To reach our goal, we will communicate more with our stakeholders in the beginning of the project to find out what the most essential user stories to complete are. The issue in this project was that we did not have a clear communication in the beginning which meant that we had to guess what they wanted, but also what we felt could be implemented in parallel.

  To make sure that our workload for a sprint is not too hectic we should not add more tasks than necessary, and instead focus on implementing a few user stories as good as we can rather than spreading ourselves too thin.

the success criteria for the team in terms of what you want to achieve within the project (this can include the application, but also your learning outcomes, your teamwork, or your effort)

- **A:** Reflecting on how we approached this question throughout the project, it is clear that the main criteria for success for our group was extremely application based. Finishing our user stories during the sprints to create the most value for our stakeholders was always our number one priority. For example focusing on implementing an interactive map using the Google map API specifically so we can place markers and show directions for stands. Another example is putting a lot of stock in finishing all tasks when it comes to starting and ending rentals and then saving those rentals to a database that only our stakeholders could see to give them information on how many rentals are being made as well as for monetary purposes.

  Along with seeing our success from our finished user stories we also had a KPI that each sprint measured to what degree each individual group member thought they achieved their respective goals. This allowed us to see if any adjustment had to be made to how we divide our user stories amongst each other.

  Not as much focus was put on measuring our success when it came to team dynamics specifically. We always had it as a criteria to be the most productive and well-functioning group we could be. We felt like the development in those areas came automatically during the process of the project. For example we realized early on that not everyone had knowledge on how to use Git, so we made it a goal to teach those unfamiliar with it through a workshop so they could use it in the project and perhaps in future ones as well. Teamwork was mainly built through continuously working together with things like pair-programming and always being communicative with each other. Effort was always there for everyone as well, no one deliberately slacked off or didn’t do their work on time.

- **B:** In future projects, more focus could be put on having criterias for success or overall goals for things like what we want to learn during the project, our teamwork and our effort towards the project. We had a cumulative goal for all of these, but smaller, more distinct goals would be more beneficial. Right now as seen above we can only discuss these things in quite broad terms. We might have improved, but how much or how well we thought it went can't really be answered right now unfortunately. Creating these goals or measurements will make it so we can assess our success in a more tangible way like we did with our user stories. By having smaller goals also makes it clearer for the members exactly where we are currently and where we are heading as a team.

- **A → B:** For this to come true in a future project, we should come together and focus early in the process on what we collectively want to achieve as a project group alongside the actual development and success of our application. Dividing our thoughts into smaller sections by raising questions like "*what do we want/need to learn as a team during the project?"*, "*how we should make sure our teamwork is always on top?"*, *"how much effort individually should we put into this project?"* etc.. Then alter our answers into tangible criteria for us to achieve.

  KPIs for measuring these things could be used as well. For example checking how well each member thought we worked as a team every sprint through some measurement like we have done with our user stories.

your user stories in terms of using a standard pattern, acceptance criteria, task breakdown and effort estimation and how this influenced the way you worked and created value

- **A:** We have learned to write user stories in terms of the standard pattern "As a X I'd like to Y so that Z". These stories have been developed together with our stakeholders and product owner, so we as a team know what they want and what to prioritise to make the most value possible.

  To make the development process easier we created task breakdown on all of our stories together with acceptance criteria. We have used our acceptance criteria as a type of task breakdown, meaning that we did not write every task down in a separate checkbox list. Our stories have also been developed with the INVEST criteria in mind, to make sure we structure them in the best way possible. The hardest part for us was to estimate the time each story would take, since we were new to android development but it became easier for every new sprint. All of our stories have also included our DoD (Definition of Done), which is a list of things that needs to be done in order to mark a user story as done.

In each sprint we assigned different team members to the stories we have planned to make sure every story gets done. The stories were either assigned to a specific member or to a couple of members to code together.

- **B:** We have learned a lot of things regarding user stories that are valuable and we will use the practices in future projects. But there are some things that we can make improvements on. We can define and structure a task breakdown to make the implementation of a user story in a more efficient way. One of the problems was that we struggled a bit in the beginning with how to prioritise the stories to make the most amount of value. We stumbled upon this due to bad communication with our stakeholders. That is definitely one major thing that we have learned and will take with us. Another thing we struggled with in the beginning was time estimation, which is an important thing to make sure we can finish the sprint in time, which we became better at the end of the project.

- **A→ B:** For the whole process we will use the same structure as we did in this project, create an epic and collect the functionality, then make stories out of them. On all of the stories we will then create acceptance criteria and an actual task breakdown, but also ensure that they follow INVEST criteria to make each sprint as efficient as possible.

  To make sure we don't make the same mistakes again we will have better communication with our stakeholders in the beginning of the project. This will help us know more precisely what our stakeholder would like to be implemented and in what order to make the most amount of value. For us to be able to estimate time better in the future we need more knowledge about the specific language, ID, framework etc. so we have a clue on how much time a specific task can take. To not make this mistake again we need to have more knowledge and be more prepared in the beginning of the project.

your acceptance tests, such as how they were performed, with whom, and which value they provided for you and the other stakeholders

- **A:** Our ambition was always to use our customers as a part of our acceptance test and therefore we mainly did customer acceptance testing. Our goal was to use acceptance tests as much as possible to get the development process verified at each step. This really helped us since we could identify errors and fix them directly when they occured.

  For this to work we had to have continuous contact with the stakeholder. We mainly had contact with them through Zoom and we shared our screen and showed them how the application looked and worked so far. This gave good value for us and allowed us to prioritize and work on what they wanted in the application rather than us implementing and working on what we might have thought they wanted. This gave

value to the stakeholders in return since we could make sure that the requirements were met.

- **B:** In future projects we would have liked to do user acceptance testing. This would allow for users to interact with the software itself in order for us to get feedback on implemented features and the overall experience. User acceptance testing is important since users are naive of how the application works. They use the application with a fresh mind and ensure that the functionality works correctly.

  Other than that we would still perform customer acceptance testing but we would like to do it in a more qualitative way. We felt that we did as good as we could considering the tough times because of covid-19. In a normal time we would not do the tests via Zoom but rather in a place where the customers could perform the tests on a physical smartphone. We think that this would give a lot more value since this is the intended way to interact with the application.

- **A → B:** How we would achieve this is to keep having a strong communication with our stakeholders to be able to perform these tests. We would perform the tests in a more structured way to make sure that everything we do is tested and delivers the intended value. This would give confidence in the system and ensure that the system is eligible to be used by the users. The users acceptance tests would be based on user stories to make sure that we meet the requirements. They could be done in an iterative process to make it easier for us as developers to make changes or fixes to the application rather than waiting until the very end when we think we have a product ready for launch.

the three KPIs you use for monitoring your progress and how you use them to improve your process

- **A:** We had a bit of a hard time to come up with some good KPIs at the start of the project. This was because we wanted something that really would suit our group to monitor our progress rather than deciding on some KPIs just because we were meant to. Since we had a lot of different talent in the group, finding the most optimal KPIs that could cover all of these bases took more time than we had originally thought. However, when we finally decided on four KPIs and could use these to measure value, they really helped us get a clear view of how our work over the four last weeks paid off and they could guide us in the right direction.

  The four KPIs we chose to use were:

  1. **Our individually experienced workload**
     - 1-5 where 1 is very calm and 5 is extremely stressful
     - This KPI helped us monitor how each member of the team felt about a sprints workload. The average score of this KPI was between 2,3 to 2,8 which feels reasonable. It was good for each of us to reflect about this

since it gave us an idea of how comfortable we were with the things we worked on as well as getting an overview if the sprints distribution of work seemed relevant.

2. **In which scale we individually completed the goals for this sprint**
    - 1-5 where 1 is total failure and 5 is all goals achieved
    - It was important for us to see how well we reached our goals, especially in comparison to how much workload we experienced. Without this KPI it would be hard to get a picture of if a week's work really gave any results or not. If this KPI would have a low score and the first KPI a high score, we would probably have to make some changes in our planning. Our score varied from 4,3 to 5,0 which we are very satisfied about.
3. **Our customers assessment of delivered value**
    - 1-100 where 1 is total failure and 100 is everything fulfilled
    - This KPI was handled by our customers so that we could get a measure of how they felt that we delivered each sprint. The lowest score we got on this was 75 after the first sprint with the use of KPIs. It was not terrible but not great either, and it's clear that we started to prioritise more important things in terms of value the following weeks since we then got an average of 89.
4. **Our own assessment of delivered value**
    - 1-100 where 1 is a disaster and 100 is could not have delivered more value
    - This KPI was used to measure how we felt about our delivery as a team in terms of value and to be able to compare this with our customers' assessment. It was a good sign if these two scores were relatively close to each other because that meant that the expectations from both parts were fairly similar.

After each sprint we evaluated our performance by filling in a sheet and we could in an easy way see how much of our goals was accomplished in comparison to how much time we put in by comparing the first and the second KPI. It was important for us that to reach our goal with the use of KPIs we had to be careful when rating our performance so we could get a result that would reflect the reality. Considering the lack of knowledge we had about using KPIs going into this project we are satisfied how our use of these KPIs turned out.

- **B:** In this project we realized how important it is to have strong KPIs in order to measure the efficiency to reach our goals. In a future project we would definitely make sure to choose KPIs in an earlier stage so that we can measure our work from the very first sprint. We were pretty satisfied with the selection of KPIs that we made but considering we would like to involve regular users of the application in the future

and not just the customers we think that it would be great if we had a KPI that would measure the users satisfaction after for example user tests.

We would also spend more time analyzing the KPIs and ask ourselves why the outcome of the KPI shows the result that it does. What was it that made us achieve/not achieve our goals so well this week? Was it how we spent our time? Was it because of the planning and structure or because of lack in communication with our stakeholders?

- **A → B:** In order to achieve this we think that it is important to sit down very early and take our time to come up with good KPIs. We would ensure that the KPIs are relevant by asking ourselves questions regarding each KPI such as "Is this easy to understand?", "Does it focus on improvement? and "Is it derived from a valid strategy?" just to name a few guidelines in order to design a good KPI.

  A way to improve the analysis of KPIs is to take a few minutes on each KPI and share our thoughts of why it had a good/bad outcome. This way it can be easier to see a clear pattern of how our way of working delivers value and we can get an idea of what we are doing right and what has to be improved. This can be done after each sprint and would help us continue working in an efficient way.

Social Contract and Effort

your social contract (Links to an external site.), i.e., the rules that define how you work together as a team, how it influenced your work, and how it evolved during the project (this means, of course, you should create one in the first week and continuously update it when the need arrives)
There is a survey (Links to an external site.) you can use for evaluating how the team is perceiving the process and if it is used by several teams it will also help you to assess if your team is following a general pattern or not.

- **A:** The group had limited experience of using a social contract before the project. Some of us had never used this before and consequently, we have learned a lot about how and why to use a social contract. We have experienced that the biggest advantage with the contract is that we planned the working process, how we would communicate etc. before the project began and thereby, we felt that the process went smoothly. As we have mentioned before, we used a template to create the first version and this was surprisingly extensive which has resulted in the final contract being very similar to the first version. However, we have added a few parts in order to make it even more personalized. For example, we added exactly how our group would use scrum-master and product owner.

To summarize, our goal was to create a comprehensive contract and to update it whenever it was needed. We feel that we achieved our goal and a major reason for this was that we were consistently discussing problems, solving these problems and finally updating the contract.

- **B:** In the future projects, we believe that we will use the complete idea of a social contract, mainly because of the reason above, the initial plan of the project the social contract provided us with. Often when you start a project you discuss how you will accomplish your goals in terms of results but rarely you discuss how you will communicate etc and this part will be brought with us until the next project. If something would be changed until the next project, it might be to discuss even more if something needs to be updated, maybe in the end of each sprint, instead of doing it when a misunderstanding occurs.

- **A → B:** How we would implement this depends on the experiences of social contracts our future group members possess. But if we assume that they haven't used a social contract before, we believe that the key is to explain all the benefits with this method. In our case, we didn't have that much experience and therefore, it was difficult to understand how beneficial a social contract is. In the next project, we will remember this project and pass on the best parts to the future group members.

the time you have spent on the course and how it relates to what you delivered (so keep track of your hours so you can describe the current situation)

- **A:** The time we have spent during the project has varied but this can simply be explained by the amount of work that we planned to do each week. In the beginning we prioritized wrong, moved too many user stories to the sprint backlog and thereby we needed to spend more time than we should, around 25 hours per person. When we got more experience, we moved a more appropriate amount of user stories and thereby, we worked around 15-18 hours per week which was sufficient for us and the project. In addition, we spent the time we needed to finish most of our user stories each week and according to our customers, we had a mean of 85 (out of 100) in delivered customer value. The single values were 75, 80, 90 and 95, sorted by week increasingly, which also shows that we learned how to work along with the experience.

- **B:** We will use our experience in future projects and above all, we will use our experience regarding equalizing the time required during each week and thereby equalizing the delivered value to the customers. This will lead to a nice and high performance during the whole project which has been the goal of the majority of all projects we have done so far and probably will be in the future as well.

- **A → B:** To make this change come true in future projects, it's important to try to detect how much time a user story requires and also discuss with customers what they expect, already in an initial stage. Thereby, the group will be able to predict how to set up their work to get a nice and productive development process during future projects.

Design decisions and product structure

how your design decisions (e.g., choice of APIs, architecture patterns, behaviour) support customer value

- **A:** Since our customer is the actual service supplier, we have been considering both the customer's value perspective and the end user value perspective. The end user's value has primarily been defined by our customer, since users are their customers. Additionally, it has been interpreted and elaborated through the use of an internal product owner. The choice of using the Google Maps API and creating an app that is centered around the map activity was based on our customer's wishes to mimic the dominant design on the market for similar applications. Furthermore, we created a navigation menu with pages requested by our customer. The corresponding activities and/or fragments to these pages were implemented in order according to value added from the customer's perspective. However, this means that some pages are yet to be implemented. In creating these pages, we have considered our customer's demands for homogeneity as well as simplicity and clarity. The internal product owner was useful in the process of deciding how best to deliver simplicity and clarity by designing an intuitive user interface and smooth user experience. The process included continuous feedback from customers regarding our design decisions. A great example of how a decision supported customer value by reflecting their wishes and demands was the choice to enable rentals, i.e. the full value creating loop, to be executed without leaving the main activity. Rentals are now initiated, viewed and finished in a window that appears on top of the map activity.
- **B:** Our customers are content with the design decisions we took and the resulting design, as are we. Although, we have been consulting the customer during several meetings throughout the project timeline. In future projects, the value of having a clear understanding of their vision and demands regarding design from the start will not be underestimated. However, continuous feedback and the flexibility to adapt accordingly will still be needed.
- **A → B:** A better understanding of the envisioned design and the customer demands in the beginning of the project could be achieved by conducting an early meeting with the purpose of discussing design decisions, co-creating mockups and ensuring that both parts are on the same page. Including customers this way could potentially make them feel included in the creation of the app. In many cases, this also leads to a higher perceived value due to the IKEA effect.

which technical documentation you use and why (e.g. use cases, interaction diagrams, class diagrams, domain models or component diagrams, text documents)

- **A:** We created a domain model early on, in order to structure our planning around a common perception of how the app would look in terms of classes. We used a domain model since it is a quick and easy way of defining the corner stones and making sure everyone has the same vision in mind going forward. As the discussions move to become more complex, the domain model is a comfortable base on which to fall back and anchor decisions. Moreover, mockups were used during the implementation. Firstly, this enables immediate constructive criticism from customers by serving as a concrete basis for discussions and consultations. Secondly, it simplifies the process of implementing the views. By following a mockup, it is easier to stay in line with the agreed upon design. Finally, a UML-diagram was made a couple of weeks into the development process. They were created when the original domain model had become a bit outdated. The creation of the diagram enabled the group to refresh vision and perception of the application structure. It was also an opportunity to sharpen the definition, by adding details and logic, as well as making some improvements. The discussions revolved a lot around how the model could become as modular and extendable as possible.
- **B:** This project taught us that mockups are very useful when working towards a customer. The lesson we will bring with us, is to create models and diagrams proactively and updating them continuously, rather than creating them reactively once the old ones have become outdated. It is easy to sway from the technical documentation and to let go of it as the development progresses, but updating it frequently and incrementally as new decisions are made will pay off when facing uncertainty.
- **A → B:** In future projects, we will correct our mistakes by creating comprehensive documentations early on and making sure it is up to date with every new decision that could impact the conceptual or logical model. Documenting the entire process will also enable faster troubleshooting and evaluation if something goes wrong in a project. In group projects like this one, the team members can remind each other to consistently either keep in line with current technical documentation or propose changes and updates to it.

how you use and update your documentation throughout the sprints

- **A:** At the start of the project, referencing our domain model alongside our mockups when developing our application gave us great early guidelines as to how the project was supposed to be built up and how it was supposed to look like. It was helpful not only for the person/pair doing a specific task for a mockup or component of our model but for the other group members as well, it gave us as individuals a better collective understanding and visualization of exactly what everyone else was working on and how it could potentially connect with what us ourselves were working on.

  During the project's lifetime we continued updating mockups for already existing pages and any eventual new pages that we had to create for the application. This helped maintain a good workflow for the group and ensured a higher-quality product for our stakeholders.

  The 5th week into our project we decided to update our domain model into a UML-diagram. We felt quite quickly that our initial model was becoming more and more inadequate as the application progressed and grew and because of this we needed a better, more in-depth way of visualizing the structure of our application. This new diagram helped us realize that we had to make our structure more modular. An example of this is that with this diagram we noticed that some logic that was in the front-end of our application, especially when handling our rentals, was supposed to be in the back-end. Because of this we had to refactor a good amount of code and re-test everything to make sure everything was working as intended again, costing us time.

- **B:** While creating a domain model allows for a great general sense of a new application for the group, nobody can precisely predict at the start of the project how an application is going to be constructed. That's why in future projects, creating a more in-depth UML-diagram earlier on in the project when the team has a greater idea of what the application is going to look like or at least updating and improving our already existing model would create much less confusion and would boost our workflow. It would also save us valuable development time as you saw above with the rental logic.

  When you look at how we worked when it came to the development of the graphical side of the application, it went much more smoothly then the logic-based side. This is because we steadily upgraded and improved our designs in the mockups and therefore everyone was on the same page on how the application looked.

- **A→ B:** For this to be attained in future projects, it is important for the group to realize the project is growing alongside us, and is not a static template for us to work off of. We realized this on the visual side of things and created a great skeleton for ourselves but we failed to do this on the logic-based side, the actual meat of the application. Therefore greater emphasis has to be put on updating our documents in that

department more regularly whenever any of the group members recognizes that changes need to be made on the structure.

how you ensure code quality and enforce coding standards

- **A:** Our code base is structured with the software architectural pattern MVVM that separates the development of the GUI (the view) from the development of the business logic or backend logic (the model) so that the view is not dependent on any model platform. Other than that we have used the singleton pattern to make it easier to use the same instance all over in the application.

  In terms of design principles we have focused on the principals to make the app extendable, maintainable, easy to read in terms of variable names and functional decomposition. In order to make the code clean we have frequently checked the naming of variables and methods, to make sure they suit its purpose. Other than that we have used the common code standards of java, the most common ones can be found here.

  To make sure our code works as intended we have made JUnit tests to find as many mistakes as possible. By implementing tests for all public methods in the backend logic (the model) we can make sure that the functions work for (most) edge cases that we are aware of, which makes our code base more maintainable when writing new code as well as helps us finding and minimizing the amount of bugs.

- **B:** We have learned the pros of a good code base with quality code in earlier courses but it becomes very clear in a project like this. In future projects we are going to use the same way to structure our codebase. Start with a software architectural pattern that fits the framework/IDE, and use principles and patterns to further build out the code base in the best way possible. An important thing to have in mind while developing a codebase is to make sure it is easy to maintain and extend in the future, which have impacted our work and is one thing that we will bring into future projects.

  We have also learned the positives of good variable/method names, to make sure it is readable and easy to understand for others. It saves time and effort for us as a team to not change names or try to understand what a method does due to a bad name.

  One thing that we will especially have in mind in future projects is to start writing good tests earlier to make sure the methods work as intended. We will partially take the quote “if it isn't tested, it's broken” with us for future projects.

- **A → B:** These things are important to bring into future projects and we will do that by starting thinking about all the practices earlier then we did in this project. When we start a new project, make sure that everyone in the team knows about all the principles you want to use. Have a specific meeting where you discuss the architecture pattern for the specific application, what principles to use and how to use them and also

which conventions to use. This will help the development, and you can save some time to not discuss and explain the structure to different team members.

Application of Scrum

the roles you have used within the team and their impact on your work

- **A:** We have learned that having roles within our team impacts our work in a very positive manner. We did not have any specific roles in the beginning of the course, which affected the meetings to be unstructured and inefficient. Later on we decided to use two specific roles, which is a scrum master and a product owner. We decided to rotate scrum master one time so that everyone that wanted to try got the chance to do it. The second scrum master became the one for the rest of the course. Our standups became shorter and we only talked about the necessary stuff. With both a scrum master and a product owner, it feels like we have prioritised the user stories better. These roles have made a good value for us as a team.

- **B:** In future projects, we want to have a scrum master and a product owner at the start of the project to ensure that we get off to a good start. We would also like to only have one scrum master so that that person evolves in that role, which enables him/her to have as much positive impact on our work as possible. All this would lead to a more efficient workflow within the team.

- **A → B:** To achieve this, the team should spare time before starting the project to decide who should be the scrum master and product owner. The team could also help out to provide useful information about the different roles.

the agile practices you have used and their impact on your work

- **A:** During the project we have used iterative development cycles(or sprints) for our application development. This includes sprint planning, sprint review, standups (2x a week) and retrospective. The sprint starts with a planning session where we discuss our objectives for this week and divides the work to be done. This includes coming up with the AC for each user story. Each person/pair works with their user story until it is done (DoD and AC complete). The next step has been to check in with our customers, giving them our deliverable application and letting them test and ascertain whether or not we have met their requested standards and criteria. Lastly, as a team we accumulate our work and review with our KPIs. Based on this, we decide whether or not our workflow/planning have to be changed.

  This process has given us a very easy to follow system of our project development, and has led to much less confusion within the group on what has to be done to ensure great structure in both our work and team-dynamic. It has also facilitated a greater

encouragement from our customers to give feedback on our work with every sprint, making it easier for us to refocus on what they actually want in the application.

Our tasks are implemented through user stories that are posted as cards on a virtual board tool called Trello. This tool has helped us visualize our workflow immensely. It allows us as a group to see where everyone is at in their process as we can move these cards through different stages of the task development, “Sprint backlog”, “In progress”, “Testing” and “Done”.

- **B:** In future projects we can take with us this specific workflow because it worked very good for us. This made us to be more structured, focused and never confused regarding what to do next. What we can improve can be to prepare the meetings better before each sprint and maybe to plan more ahead of us so that we all have standing deadlines. We could also as a group internally have a more effective way of reviewing our work from the start. This was done towards the end using pull requests.

- **A → B:** To achieve this, we can introduce this way of working before the project starts to ensure that everyone is on the same foot from start. We can also provide the team with useful documentation and talk about our previous experiences and how to avoid/implement it all depending if it was a good or bad one. This ensures that we will have good agile practices and that it will have a positive impact on our workflow in future projects.

the sprint review and how it relates to your scope and customer value (Did you have a PO, if yes, who?, if no, how did you carry out the review? Did the review result in a re-prioritisation of user stories? How did the reviews relate to your DoD? Did the feedback change your way of working?)

- **A:** There are a lot of things that we have learned during this project. The scrum lifecycle is one of them, but especially the sprints reviews. In this project we have learned all the pros of a well structured sprint review, and how that can improve the productivity and the value for the stakeholders.

  During this project we have had Theo as the product owner. In our sprint reviews the attendees was the team, scrum master, product owner and our stakeholders. The review consisted of a demo of the app for our stakeholders, to show them what we have accomplished and compared the product to what we planned in the beginning of the sprint. When the demo was finished the word was given to the stakeholders for feedback on the progress. Later on the product owner presented the product backlog for the stakeholders to get feedback for upcoming sprints, to make the most value possible going forward. The feedback we got from the stakeholders resulted often in a re-prioritisation of user stories in the backlog but also the way we worked, e.g. the pairs we worked in or which part of the app each member should work with. A specific thing the feedback sessions helped us with was how we reviewed our code,

which ended up being with pull-requests. This was done to get a more readable codebase and easier to maintain, it was also added to our Definition of Done (DoD).

However this was not the way we worked in the beginning, we did not have a very close connection with the stakeholders which ended up with a bad prioritisation and a bad product at the end of the sprint, and it didn't give the stakeholder any good value.

- **B:** For future projects we will use the structure that we have used, or at least have it in mind when we structure up that specific project. We will let a Product owner run the meetings, have good communication with the stakeholders and prioritise the backlog after their feedback to develop the most value possible. But also to see how the stakeholders' feedback can improve the workflow for us as a team, which can include how we as a team work together or how a specific user story should be done, e.g. add a task to the Definition of Done (DoD).
- **A → B:** To make this come true we will in the beginning of the project assign a product owner and get his thoughts about the project to prioritise the backlog the way we think can make the most value possible. Later on have good communication with the stakeholders to get their thoughts on our priorisation, and if they have any other ideas on how the work should be done.

best practices for learning and using new tools and technologies (IDEs, version control, scrum boards etc.; do not only describe which tools you used but focus on how you developed the expertise to use them)

- **A:** The very first tool we knew we had to use was Trello, in order to have a convenient way of writing down our user stories and applying them to group members. Since this was not the first project course in programming for many of us, we already knew how to write user stories and how to use Trello. Therefore, we felt like there was not a need to actually read more about it.

  For version control, we used git with the help of GitHub and GitKraken for a simplified way to interact with our project. A workshop was held for everyone that was not very confident about git, and this gave everyone the basic knowledge to move forward and start with the development.

  We used Android Studio for most of our app development, this includes Java and XML. We developed our expertise to use them by making sure that in the beginning of the course, the group members who had experience of using it before were programming with someone who had less experience. To actually learn how to use tools and frameworks inside of Android Studio, YouTube was our main source of inspiration and Stackoverflow whenever we had minor issues with bugs.

  Later on in the project, our stakeholders asked us to add rentals in a database for them to be able to fetch easily. Some of us had stumbled upon Firebase previously and we knew that there were many resources on the web for it as well as simple syntax to

connect to it. To learn it, one of us dug deeper into it by watching tutorials on YouTube as well as Google's own documentation of it. The most basic features of the database were then implemented and a workshop was held for the rest of the group to learn how to use it.

In the beginning of the project we made some mockups of the app, for the team to know what the app should look like and to simplify the development process. This was made with Adobe XD, which is a UI/UX design tool developed by Adobe Inc. Some of the team members have worked with the software before and had some confidence using it, some others learned the basics using YouTube videos.

- **B:** Our knowledge of these technologies will of course be used in future projects. Trello will probably be used directly as it is an easy and intuitive way to structure user stories. Git and GitHub is the most common version control system and it is definitely something we will stumble upon in the future. Some of the group members had never used this tool before, and it was definitely more of a challenge for them getting started. However, after a few weeks, everyone got more efficient at pushing, pulling and merging. Pull requests were new to everyone and it could be used more in future development, especially in the beginning of the project, as it really helps maintain code quality.

  Android Studio is very similar to other environments we have used, and the experience we picked up from using it in this course will of course help us in future object-oriented programming projects, especially if we will be doing more app development since there are many similarities with iOS and Android. Coding in Java has enhanced our skills in object-oriented programming, including syntax, structuring up the code and using Google APIs. This is something that we believe can be helpful to know a lot about in any software project.

  The go to strategy for learning new technologies as quickly as possible was definitely to have someone with more knowledge teach others with less experience. This is because it allows for beginners to get the basics in practically no time, while the person who is responsible for the workshop usually also learns new things when teaching. Some other useful strategies that we have used are documentation, forums (e.g. Stack Overflow) and youtube tutorials.

- **A → B:** To make the most out of the technologies we have used in this course in upcoming projects, we should look back at how we used them and take inspiration from this project. To further expand our knowledge for the different tools, we should consider having workshops whenever possible, since it proved for at least our group to be the most effective way of learning new technologies.

relation to literature and guest lectures (how do your reflections relate to what others have to say?)

- **A:** In the beginning we tried to apply the way we used scrum through the literature and lectures given without thinking some things through. We did not have a scrum master or anyone who held in the meeting, which made it inefficient. Later on we got very useful constant feedback by our supervisor who gave us tips on how to make it run more fluently. This included, for example, to introduce a scrum master. We have also been getting useful tips such as prioritizing our user stories to deliver as much value as possible for our stakeholders. With this being said, we have made very good improvements based on reflections to what others said. We have always been open minded and hungry for new knowledge within the agile workflow. This implies that we always at least tried to apply new things that others said, mainly from our supervisor. We feel like the whole group worked towards making the most out of this course and really tried their best individually to come up with ideas and provide useful documentation/material. Oskar Nyqvist said during the guest lecture that we should allow ourselves to try out new things and to make mistakes. This would then lead to us reflecting whether or not we should continue doing that new thing. We have followed this rule of thumb during the whole project, which enabled us to always reflect on the things we were doing to come up with new better solutions. One example of us doing this was with our standups where we decreased the time and the thoroughness of everyone's work. This implied the meetings to be shorter and much more efficient.
- **B:** In future projects, we should take with us this experience that we have gained throughout this course. Something that we always can do is to develop and polish our agile process. What we all could possibly do is to freshen our memory regarding every part of a scrum lifecycle to make sure that we do not forget anything essential. We have come to the conclusion that it is very important to listen to what others have to say and to do reflections. This enables us to actively look for better solutions to what is already working. This implies being open minded and optimistic regarding new ideas to increase the workflow efficiency or to increase the cohesion inside the team. It would also be good to take help from day one from a supervisor that sees the workflow from a different angle.
- **A → B:** How we would develop and polish our agile process would potentially be to investigate and do deeper research how different companies implement the process. This would enable us to actively look for better solutions that fits us better as a team, which is just being open minded. The key would be to always have that workflow efficiency in mind and to reflect on every process, both in the start of the project and during it. This would bring an incremental process of maximizing efficiency. There would also be a good thing to get a hold of an outsider supervisor that has nothing to do with the project. This person could bring a lot of ideas and new intervention to the team.